PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-207224
(43)Date of publication of application : 28.07.2000

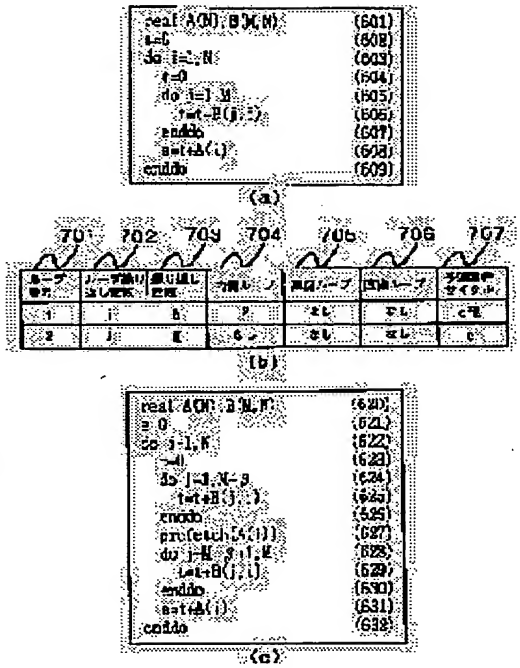
(51)Int.Cl. G06F 9/45
G06F 12/08

(21)Application number : 11-008234
(22)Date of filing : 14.01.1999

(71)Applicant : HITACHI LTD
(72)Inventor : KUSHIMA ICHIRO

(54) SOFTWARE PREFETCHING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To prevent prefetched data from being expelled from a cache before being used, when data which is not in the innermost loop is referred to.
SOLUTION: A loop (605 to 607) immediately prior to array reference A (i) is taken out, when a number (d) of predictive execution cycles of the whole loop (605 to 607) is larger than a number (k) (memory latency) of cycles needed to prefetch data, a number (e) of predictive execution cycles per loop (605 to 607) is found, and a loop frequency β=(k/e) corresponding to the memory latency is calculated. Then the loop (605 to 607) is divided into two loops, i.e., a former half loop (624 to 627) for repetition of 1 to (M-β) times and a latter half loop (628 to 630) for repetition of (M-β+1) to M times and a prefetch instruction (627) is put between the former half loop (624 to 627) and the latter half loop (628 to 630).



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－207224
(P2000－207224A)
(43)公開日　平成12年７月28日(2000.7.28)

(51)Int.Cl.'	識別記号	ＦＩ		テーマコード'(参考)
・Ｇ０６Ｆ　9/45		Ｃ０６Ｆ　9/44	３２２Ｃ	５Ｂ００５
12/08		12/08	Ｄ	５Ｂ０８１

審査請求　未請求　請求項の数12　ＯＬ　（全 13 頁）

(21)出願番号　特願平11－8234

(22)出願日　平成11年１月14日(1999.1.14)

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　久島　伊知郎
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内
(74)代理人　100068504
弁理士　富田　和子

Ｆターム(参考)　5B005 JJ13 MM01
5B081 CC30 CC41

(54)【発明の名称】　ソフトウェアプリフェッチ方法

(57)【要約】
【課題】　最内側ループ中にないデータの参照に対して、プリフェッチしたデータが使用される前にキャッシュから追い出されるのを防止する。
【解決手段】　配列参照Ａ（ｉ）の直前にあるループ（６０５～６０７）を取り出し、ループ（６０５～６０７）全体の予測実行サイクル数ｄがデータをプリフェッチするのに必要なサイクル数（メモリレイテンシ）ｋより大きければ、ループ（６０５～６０７）のループ１回あたりの予測実行サイクル数ｅを求め、メモリレイテンシに相当するループ回数β＝［ｋ／ｅ］を計算する。そして、ループ（６０５～６０７）を、１～（Ｍ－β）回の繰り返しを実行する前半ループ（６２４～６２７）と、（Ｍ－β＋１）～Ｍ回の繰り返しを実行する後半ループ（６２８～６３０）の２つのループに分割し、前半ループ（６２４～６２７）と後半ループ（６２８～６３０）の間に、プリフェッチ命令（６２７）を挿入する。

図5

```
real A(N),B(M,N)          (601)
s=0                       (602)
do i=1,N                  (603)
  t=0                     (604)
  do j 1,M                (605)
    t=t+B(j,i)            (606)
  enddo                   (607)
  s=t+A(i)                (608)
enddo                     (609)
```

(a)

701 702 703 704 705 706 707

ループ番号	ループ繰り返し変数	繰り返し回数	内側ループ	二重ループ	直後ループ	予測実行サイクル
1	i	N	2	なし	なし	c*M
2	j	M	なし	なし	なし	c

(b)

```
real A(N),B(M,N)          (620)
s=0                       (621)
do i=1,N                  (622)
  t=0                     (623)
  do j=1,M-β              (624)
    t=t+B(j,i)            (625)
  enddo                   (626)
  prefetch(A(i))          (627)
  do j=M-β+1,M            (628)
    t=t:B(j,i)            (629)
  enddo                   (630)
  s=t+A(i)                (631)
enddo                     (632)
```

(c)

【特許請求の範囲】
【請求項1】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおいて、
プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、
当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数βを求め、
前記第2のループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割し、
前半ループと後半ループの間にプリフェッチ命令を挿入することを特徴とするソフトウェアプリフェッチ方法。
【請求項2】 プリフェッチ命令を持つプロセッサに対するオブジェクコードを生成するコンパイラにおいて、
プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、
当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数βを求め、
前記第2のループの、最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入することを特徴とするソフトウェアプリフェッチ方法。
【請求項3】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおいて、
プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、
当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求め、
前記ループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割し、
前半ループと後半ループの間にプリフェッチ命令を挿入することを特徴とするソフトウェアプリフェッチ方法。
【請求項4】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおいて、
プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、
当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求め、
前記ループの、最後からβ回目の繰り返しのときに、プリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入することを特徴とするソフトウェアプリフェッチ方法。
【請求項5】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置であってプリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数βを求める手段と、
前記第2のループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割する手段と、
前半ループと後半ループの間にプリフェッチ命令を挿入する手段とを備えることを特徴とするコンパイル装置。
【請求項6】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置であって、
プリフェッチの対象とするメモリ参照を含む第1のループがその内側に第2のループを有する場合に、当該第2のループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第2のループの繰り返し回数βを求める手段と、
前記第2のループの、最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入する手段とを備えることを特徴とするコンパイル装置。
【請求項7】 プリフェッチ命令を持つプロセッサに対するオブジェクコードを生成するコンパイル装置であって、
プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、当該ループのループ繰り返し1回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求める手段と、
前記ループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割する手段と、
前半ループと後半ループの間にプリフェッチ命令を挿入する手段とを備えることを特徴とするコンパイル装置。
【請求項8】 プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置であっ

て、
プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、当該ループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求める手段と、
前記ループの最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入する手段とを備えることを特徴とするコンパイル装置。
【請求項９】　ソースコードを読み込んで、プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイル装置において、
前記ソースコードが、第１のループを有し、更に、当該第１のループの内側にプリフェッチの対象とされるメモリ参照及び第２のループを有する場合、
前記オブジェクトコードは、
前記第２のループに対応する前半ループ及び後半ループと、
前半ループと後半ループの間にプリフェッチ命令とを有し、
前記後半ループの繰り返し回数は、データをプリフェッチするのに要するサイクル数に相当する回数であることを特徴とするコンパイル装置。
【請求項１０】　プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラであって、
プリフェッチの対象とするメモリ参照を含む第１のループがその内側に第２のループを有する場合に、当該第２のループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第２のループの繰り返し回数βを求めるステップと、
前記第２のループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割するステップと、
前半ループと後半ループの間にプリフェッチ命令を挿入するステップとを備えることを特徴とするコンパイラを記録した記録媒体。
【請求項１１】　オブジェクトコードを記録した記録媒体であって、
前記オブジェクトコードは、
１つのループ処理に相当する前半ループ及び後半ループと、
後半ループのうしろに、メモリ参照をする命令と、
前半ループと後半ループの間に、前記メモリ参照についてのプリフェッチ命令とを有し、
前記後半ループの繰り返し回数は、前記オブジェクトコードを実行可能なコンピュータにおいて、データをプリフェッチするのに要するサイクル数に相当する回数であることを特徴とする記録媒体。
【請求項１２】　オブジェクトコードを記録した記録媒体であって、
前記オブジェクトコードは、
ループと、
当該ループのうしろに、メモリ参照をする命令と、
前記ループの最後からβ回目の繰り返しのときに、前記メモリ参照についてのプリフェッチ命令を実行するコードとを有し、
前記βは、前記オブジェクトコードを実行可能なコンピュータにおいて、データをプリフェッチするのに要するサイクル数に相当する前記ループの繰り返し回数であることを特徴とする記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、計算機利用技術におけるコンパイル方法に関し、特に、プリフェッチ命令をコードに挿入することによりキャッシュミスの削減を図るソフトウェアプリフェッチ方法に関する。
【０００２】
【従来の技術】近年、命令レベルの並列度や動作周波数の向上によってマイクロプロセッサの性能は飛躍的に向上している。これに対して、計算機の主記憶を構成するＤＲＡＭの性能向上はプロセッサ性能の向上に比べて低いレベルに留まっている。この結果、主記憶参照に要するサイクル数（メモリレイテンシ）は増加する傾向にある。
【０００３】これに対して、従来のシステムでは、データ参照の局所性に着目して、主記憶と比較して高速な少容量のメモリ（キャッシュ）をプロセッサと主記憶の間に配置し、最近参照したデータをキャッシュ上に置くことによって、主記憶参照の回数を減らし、全体として主記憶参照に要する時間を短縮するという方式をとっていた。ところが、数値計算処理などの大規模なデータを使用する計算では、データ参照の局所性が低いためキャッシュミスが多発し、主記憶参照によって生じる待ち時間によってプログラムの性能が大幅に低下してしまうという問題があった。
【０００４】このような大規模データに対するキャッシュミスの増加に対処するため、例えば、文献「T.C.Mowry 他,"Design and Evaluation of a Compiler Algorithms for Prefetching", Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.62-73, 1992」に記載されているように、主記憶からキャッシュへデータを先行的に読み出す特別な命令（プリフェッチ命令）をプロセッサの命令セットに用意し、コンパイラによってプログラム中にプリフェッチ命令を挿入す

る方式（ソフトウェアプリフェッチ）が検討されている。このプリフェッチ命令を利用すれば、後続のループ繰り返し等で使用するデータを予め主記憶からキャッシュへ読み込みながら、同時に別の演算を行うことができ、これによって主記憶参照による待ち時間を隠蔽することができる。

【０００５】図１１は、従来のソフトウェアプリフェッチ方法の例を示す図である。同図（ａ）は、コンパイル処理の対象となるＦＯＲＴＲＡＮで記述されたソースプログラムを表す。

【０００６】ここでは、同図（ａ）に示すプログラムにおいて、配列要素Ａ（ｉ）の参照（５０４）について、プリフェッチ命令を挿入する場合を考える。同図（ａ）に示すように、ｉ番めのループ繰り返しで配列要素Ａ（ｉ）を参照するようなループ（５０３～５０５）があった場合、まず、ループの１回の繰り返しにかかる実行サイクル数ｃを見積もり、データをメモリからキャッシュに移動するのに要するサイクル数（メモリレイテンシ）Ｌをｃで割った値α=［Ｌ／ｃ］（ここで、［］は、小数点以下切り上げを表す記号）を計算する。そして、ｉ回目の繰り返しで配列要素Ａ（ｉ＋α）をプリフェッチする命令を配列要素Ａ（ｉ）の直前に挿入する。

【０００７】同図（ｂ）は、プリフェッチ命令（５０９）を挿入した後のプログラムをソースコードイメージで表したものである。このようにα回後の繰り返しで参照するデータに対して予めプリフェッチ命令を発行しておくことにより、Ｌ（＝ｃ＊α）サイクル後にそのデータを参照するときは当該データが既にキャッシュに到着しているので、キャッシュヒットとなる。これによりキャッシュミスによるプログラム実行性能の低下が防止できる。

【０００８】

【発明が解決しようとする課題】上記文献に示されているソフトウェアプリフェッチ方法では、プリフェッチしようとする配列の参照が最内側ループ（すなわち、その内側に他のループを含まないループ）にあるような単純な場合のみを仮定している。

【０００９】前述したソフトウェアプリフェッチ方法を、プリフェッチの対象とする配列参照を有するループの内側にさらに別のループがある場合に適用すると、プリフェッチしたデータが、内側ループの実行によりキャッシュから追い出されてしまう恐れがある。

【００１０】図１２は、このような場合のソフトウェアプリフェッチ方法の例を示す図である。同図（ａ）は、コンパイル処理の対象となるＦＯＲＴＲＡＮで記述されたソースプログラムを示す。同図（ａ）に示したプログラムにおいて、配列参照Ａ（ｉ）（６０８）をプリフェッチの対象とした場合、同図（ｂ）のようにプリフェッチ命令（６１７）が挿入される。つまり、ループのｉ番めの繰り返しで配列要素Ａ（ｉ＋α）をプリフェッチしている。しかし、このプリフェッチ命令の発行の後、次以降のｉの繰り返しで、内側ループ（６１４～６１６）の実行がされるので、そこで参照されるデータＢ（ｊ，ｉ）と、データＡ（ｉ＋α）との間でキャッシュラインの競合が発生すると、プリフェッチによってキャッシュにロードされたデータＡ（ｉ＋α）がキャッシュから追い出されてしまい、実際に値を使用するときはキャッシュミスを起こすことになり、プリフェッチの効果がなくなることになる。

【００１１】本発明の目的は、最内側ループ中にないデータの参照に対して、プリフェッチの効果を得ることができるソフトウェアプリフェッチ方法を提供することにある。

【００１２】

【課題を解決するための手段】本発明に係るソフトウェアプリフェッチ方法は、プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラにおけるソフトウェアプリフェッチ方法である。そして、プリフェッチの対象とするメモリ参照を含む第１のループがその内側に第２のループを有する場合に、当該第２のループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第２のループの繰り返し回数βを求め、前記第２のループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割し、前半ループと後半ループの間にプリフェッチ命令を挿入することを特徴とする。

【００１３】この場合、プリフェッチ命令とメモリ参照との間に実行される第２のループは、β回しか回らず、βは一般に分割前の元のループ繰り返し回数に比べて小さいので、その間に参照されるデータは一般に小さく、プリフェッチしたデータがキャッシュから追い出される可能性は少なくなる。また、ループのβ回の実行サイクル数は、メモリレイテンシ、すなわち、データをメモリからプリフェッチをするのに要するサイクル数に相当するので、データを参照するときにプリフェッチデータの到着が間に合っていないということがない。

【００１４】なお、第２のループが多重ループになっている場合には、第２のループを分割してできた後半ループの内側ループをさらに分割することにより、より適当な位置にプリフェッチ命令を挿入するようにしてもよい。

【００１５】また、本発明に係る第２のソフトウェアプリフェッチ方法は、プリフェッチの対象とするメモリ参照を含む第１のループがその内側に第２のループを有する場合に、当該第２のループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第２のループの繰

り返し回数βを求め、前記第２のループの最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入することを特徴とする。

【００１６】また、本発明に係る第３のソフトウェアプリフェッチ方法は、プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、当該ループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求め、前記ループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割し、前半ループと後半ループの間にプリフェッチ命令を挿入することを特徴とする。

【００１７】また、本発明に係る第４のソフトウェアプリフェッチ方法は、プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、当該ループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求め、前記ループの最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入することを特徴とする。

【００１８】本発明に係るコンパイル装置は、プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成する装置である。そして、プリフェッチの対象とするメモリ参照を含む第１のループがその内側に第２のループを有する場合に、当該第２のループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第２のループの繰り返し回数βを求める手段と、前記第２のループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割する手段と、前半ループと後半ループの間にプリフェッチ命令を挿入する手段とを備えることを特徴とする。

【００１９】また、本発明に係る第２のコンパイル装置は、プリフェッチの対象とするメモリ参照を含む第１のループがその内側に第２のループを有する場合に、当該第２のループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第２のループの繰り返し回数βを求める手段と、前記第２のループの最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入する手段とを備えることを特徴とする。

【００２０】また、本発明に係る第３のコンパイル装置は、プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、当該ループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求める手段と、前記ループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割する手段と、前半ループと後半ループの間にプリフェッチ命令を挿入する手段とを備えることを特徴とする。

【００２１】また、本発明に係る第４のコンパイル装置は、プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、当該ループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求める手段と、前記ループの最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入する手段とを備えることを特徴とする。

【００２２】本発明に係る第５のコンパイル装置は、ソースコードが、第１のループを有し、更に、当該第１のループの内側に、プリフェッチの対象とされるメモリ参照及び第２のループを有する場合、前記第２のループに対応する前半ループ及び後半ループと、前半ループと後半ループの間にプリフェッチ命令とを有し、前記後半ループの繰り返し回数は、データをプリフェッチするのに要するサイクル数に相当する回数であるオブジェクトコードを生成することを特徴とする。

【００２３】また、本発明に係る第６のコンパイル装置は、ソースコードが、第１のループを有し、更に、当該第１のループの内側に、プリフェッチの対象とされるメモリ参照及び第２のループを有する場合、前記第２のループの最後からβ回の繰り返しのときにプリフェッチ命令を実行するコードを有し、前記βは、データをプリフェッチするのに要するサイクル数に相当する第２のループの繰り返し回数であるオブジェクトコードを生成することを特徴とする。

【００２４】また、本発明に係る第７のコンパイル装置は、ソースコードが、プログラム手続きを有し、更に、当該プログラム手続き内に、プリフェッチの対象とするメモリ参照及びループを有する場合、前記ループに対応する前半ループ及び後半ループと、前半ループと後半ループの間にプリフェッチ命令とを有し、前記後半ループの繰り返し回数は、データをプリフェッチするのに要するサイクル数に相当する回数であるオブジェクトコードを生成することを特徴とする。

【００２５】また、本発明に係る第８のコンパイル装置は、ソースコードが、プログラム手続きを有し、更に、当該プログラム手続き内に、プリフェッチの対象とする

メモリ参照及びループを有する場合、前記ループの最後からβ回目の繰り返しのときに、プリフェッチ命令を実行するコードを有し、前記βは、データをプリフェッチするのに要するサイクル数に相当する前記ループの繰り返し回数であるオブジェクトコードを生成することを特徴とする。

【００２６】本発明に係る第１の記録媒体は、プリフェッチ命令を持つプロセッサに対するオブジェクトコードを生成するコンパイラを記録した記録媒体である。そして、当該コンパイラは、プリフェッチの対象とするメモリ参照を含む第１のループがその内側に第２のループを有する場合に、当該第２のループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第２のループの繰り返し回数βを求めるステップと、前記第２のループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割するステップと、前半ループと後半ループの間にプリフェッチ命令を挿入するステップとを備えることを特徴とする。

【００２７】また、本発明に係る第２の記録媒体は、第２のコンパイラを記録した記録媒体である。この第２のコンパイラは、プリフェッチの対象とするメモリ参照を含む第１のループがその内側に第２のループを有する場合に、当該第２のループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する第２のループの繰り返し回数βを求めるステップと、前記第２のループの最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入するステップとを備えることを特徴とする。

【００２８】また、本発明に係る第３の記録媒体は、第３のコンパイラを記録した記録媒体である。この第３のコンパイラは、プリフェッチの対象とするメモリ参照を含むプログラム手続き内にループがある場合に、当該ループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求めるステップと、前記ループを、終わりのβ回を実行する後半ループと、その前までの繰り返しを実行する前半ループとに分割するステップと、前半ループと後半ループの間にプリフェッチ命令を挿入するステップとを備えることを特徴とする。

【００２９】また、本発明に係る第４の記録媒体は、第４のコンパイラを記録した記録媒体である。この第４のコンパイラは、プリフェッチの対象とするメモリ参照を有するプログラム手続き内にループが含まれる場合に、当該ループのループ繰り返し１回あたりの実行サイクル数と、データをプリフェッチするのに要するサイクル数であるメモリレイテンシとに基づいて、当該メモリレイテンシに相当する前記ループの繰り返し回数βを求めるステップと、前記ループの最後からβ回目の繰り返しのときにプリフェッチ命令を実行する、条件付きのプリフェッチ命令実行コードを挿入するステップとを備えることを特徴とする。

【００３０】本発明に係る第５及び第６の記録媒体は、オブジェクトコードを記録した記録媒体である。そして、第５の記録媒体において、前記オブジェクトコードは、１つのループ処理に相当する前半ループ及び後半ループと、後半ループのうしろにメモリ参照をする命令と、前半ループと後半ループの間に、前記メモリ参照についてのプリフェッチ命令とを有し、前記後半ループの繰り返し回数は、前記オブジェクトコードを実行可能なコンピュータにおいて、データをプリフェッチするのに要するサイクル数に相当する回数であることを特徴とする。

【００３１】また、第６の記録媒体において、前記オブジェクトコードは、ループと、当該ループのうしろに、メモリ参照をする命令と、前記ループの最後からβ回目の繰り返しのときに、前記メモリ参照についてのプリフェッチ命令を実行するコードとを有し、前記βは、前記オブジェクトコードを実行可能なコンピュータにおいて、データをプリフェッチするのに要するサイクル数に相当する前記ループの繰り返し回数であることを特徴とする。

【００３２】なお、前記記録媒体は、コンピュータが読みとり可能な任意の記録媒体であり、例えば、半導体メモリ、フロッピーディスク、ＣＤ－ＲＯＭ、磁気ディスク、光磁気ディスク等が該当する。

【００３３】

【発明の実施の形態】以下、本発明の実施の形態について図面を参照して詳細に説明する。

【００３４】図１は、本発明によるコンパイラが稼動する計算機システムの構成を示す図である。同図に示すように、本計算機システムは、ＣＰＵ１０１、ディスプレイ装置１０２、キーボード１０３、主記憶装置１０４、及び外部記憶装置１０５を備える。

【００３５】ＣＰＵ１０１は、主記憶装置１０４にロードされたコンパイラ１０８を実行することにより、コンパイル処理を行う。

【００３６】ディスプレイ装置１０２は、例えば、ＣＲＴ表示装置や液晶表示装置等で構成され、コンパイルの終了やエラー等をユーザに知らせる各種メッセージ等を表示する。

【００３７】キーボード１０３は、ユーザが、ソースプログラム１０６のコーディングを行ったり、コンパイラ１０８を起動するコマンドを入力する際等に使われる入力装置である。

【００３８】主記憶装置１０４は、例えば、ＤＲＡＭ等の半導体メモリで構成され、コンパイラ１０８や、コンパイル処理過程で必要となる中間コード１０９およびループ表１１０等を格納する。

【００３９】外部記憶装置１０５は、例えば、磁気ディスク等で構成され、ソースプログラム１０６、オブジェクトプログラム１０７等を格納する。

【００４０】図１に示したような計算機システムにおいて、ユーザがキーボード１０３等を使ってコンパイル処理の実行を指示すると、コンパイラ１０８は、ソースプログラム１０６を読み込んで、コンパイル処理を行い、対応するオブジェクトプログラム１０７を生成する。

【００４１】図２は、コンパイラ１０８のコンパイル処理の流れを示すフローチャートである。コンパイラ１０８は、まず、構文解析処理を行う（Ｓ２０１）。ここでは、ソースプログラム１０６を読み込み、字句解析、構文解析、意味解析等を行って、コンパイラ内部で処理可能な中間コード１０９を作成する。構文解析処理の詳細については、例えば、「エイホ、セシィ、ウルマン著：コンパイラＩ（サイエンス社、１９９０年）」の３０頁～７４頁に記載されている。

【００４２】次に、コンパイラ１０８はループ解析処理を行う（Ｓ２０２）。ここでは、プログラムに含まれるループの集合や各ループの間の関係等が求められ、ループ表１１０に記録される。ループ表１１０に記録された情報は、後のコンパイル処理において適宜参照される。ループ解析処理の詳細については、例えば、「エイホ、セシィ、ウルマン著：コンパイラII（サイエンス社、１９９０年）」の７３４頁～７３７頁に記載されている。

【００４３】次に、コンパイラ１０８は、各ループに対して、ソフトウェアプリフェッチ処理を行う（Ｓ２０３～Ｓ２０５）。そのため、まず、未処理のループがあるか否かを調べ（Ｓ２０３）、未処理のループがあれば（Ｓ２０３：ｙｅｓ）、未処理のループを１つ取り出して、処理対象ループｎとする（Ｓ２０４）。そして、処理対象ループｎ中の配列参照に対してソフトウェアプリフェッチ処理を行う（Ｓ２０５）。このソフトウェアプリフェッチ処理の詳細については後述する。ソフトウェアプリフェッチ処理が終了すると、他の未処理ループについての処理に進む（Ｓ２０３）。

【００４４】そして、すべてのループについて処理を終了すると（Ｓ２０３：ｎｏ）、オブジェクトコードを生成し（Ｓ２０６）、コンパイル処理を終了する。オブジェクトコード生成処理の詳細については、例えば、前記「コンパイラII」の６２４頁～７０７頁に記載されている。

【００４５】図３は、前述したステップＳ２０５におけるソフトウェアプリフェッチ処理の流れを示す図である。ここでは、処理対象ループｎの中の配列参照に対して、適切な位置にプリフェッチ命令を挿入する処理を行う。

【００４６】まず、処理対象ループｎ中に未処理かつプリフェッチの対象とすべき配列参照があるか否かを調べる（Ｓ３０１）。ある配列参照をプリフェッチの対象とすべきか否かは、例えば、その配列参照のメモリアドレスを予め求めることができるか否か、配列の添字がループ制御変数の線形関数となっているか否か、配列データが既にキャッシュにあるか否か、等に基づいて判断する。

【００４７】その結果、未処理かつプリフェッチの対象とすべき配列参照があれば（Ｓ３０１：ｙｅｓ）、その配列参照を１つ取り出して、処理対象配列参照ｓとする（Ｓ３０２）。次に、変数Ｌに、メモリレイテンシ（プリフェッチに要するサイクル数）を代入する（Ｓ３０３）。Ｌに代入する値は、ターゲットマシンによって、適当な値がコンパイラに予め与えられるが、コンパイラ起動時等に、ユーザがパラメータとして与えるようにしてもよい。

【００４８】次に、処理対象ループｎが最内側ループであるか否かを調べる（Ｓ３０４）。処理対象ループｎが最内側ループであるか否かは、例えば、ループ表１１０を参照して調べる。その結果、処理対象ループｎが最内側ループであれば（Ｓ３０４：ｙｅｓ）、従来と同様の方法で、プリフェッチ命令の挿入を行う（Ｓ３０５、Ｓ３０６）。すなわち、まず、処理対象ループｎのループ１回あたりの予測実行サイクル数ｃを求め、データをプリフェッチするのに必要なサイクル数（メモリレイテンシ）Ｌに相当するループ回数α＝［Ｌ／ｃ］を計算する（Ｓ３０５）。そして、処理対象配列参照ｓの添字式中のループ繰り返し変数ｉをｉ＋αに置き換えた配列要素をプリフェッチする命令を生成し、処理対象配列参照ｓの直前に挿入する（Ｓ３０６）。

【００４９】一方、処理対象ループｎが最内側ループでない場合は（Ｓ３０４：ｎｏ）、従来とは異なる方法で、処理対象配列参照ｓのプリフェッチ命令を挿入すべき位置を求める処理を行う（Ｓ３０７）。このステップＳ３０７における処理の詳細については、後述する。プリフェッチ命令を挿入すべき位置が求まると、その位置に、処理対象配列参照ｓをプリフェッチする命令を挿入する（Ｓ３０８）。

【００５０】以上の処理Ｓ３０１～Ｓ３０８を繰り返して、すべてのプリフェッチ対象となる配列参照について処理を終了すると（Ｓ３０１：ｎｏ）、ソフトウェアプリフェッチ処理を終了する。

【００５１】次に、ステップＳ３０７におけるプリフェッチ命令を挿入すべき位置を求める処理について詳細に説明する。図４は、ステップＳ３０７における処理の流れを示す図である。ここでは、処理対象配列参照ｓのＬサイクル前に実行される位置を求める。その際、必要に応じてループの分割を行う。

【００５２】まず、処理対象ループｎ中の処理対象配列参照ｓの直前にあるループを分割対象ループｍとして取り出す（Ｓ４０１）。ここで、処理対象配列参照ｓの前にループが存在しない場合は、処理対象ループｎ中の最後のループを分割対象ループｍとして取り出す。なお、ここでの処理対象ループｎは、最内側ループではないので、処理対象ｎループ中には、必ずループが存在する。

【００５３】次に、変数ｋの初期値として、Ｌの値を代入する（Ｓ４０２）。そして、分割対象ループｍの全体（全繰り返し）の予測実行サイクル数を求め、それをｄとする（Ｓ４０３）。分割対象ループｍの繰り返し数がコンパイル時に不明で、全体の予測実行サイクル数が求まらないときは、ｄ←∞（例えば、ｄとしてとりえない特定の値）としておく。

【００５４】次に、ｄがｋより大きいか否かを調べる（Ｓ４０４）。その結果、ｄがｋ以下の場合は（Ｓ４０４：ｎｏ）、現在の分割対象ループｍの分割は行わず、分割対象ループｍを、現在の分割対象ループｍの直前のループに変更し、また、ｋから現在の分割対象ループｍの全体の予測実行サイクル数ｄを減算する（Ｓ４１０）。なお、分割対象ループｍの前にループがない場合は、処理対象ループｎ中の最後のループを新たな分割対象ループｍとして選択する。そして、ステップＳ４０３～Ｓ４０４の処理を再度行う。

【００５５】一方、ｄがｋより大きければ（Ｓ４０４：ｙｅｓ）、分割対象ループｍの分割を行うため、まず、分割対象ループｍのループ１回分の予測実行サイクル数ｅを求め、ｋに相当する分割対象ループｍのループ回数β＝［ｋ／ｅ］を計算する（Ｓ４０５）。

【００５６】そして、分割対象ループｍを、最後のβ回のみを実行する後半ループｍ２と、その前までの繰り返しを実行する前半ループとに分割する（Ｓ４０６）。すなわち、分割対象ループｍの繰り返し回数をＮとしたとき、１～（Ｎ－β）回の繰り返しを実行する前半ループｍ１と、（Ｎ－β＋１）～Ｎ回の繰り返しを実行する後半ループｍ２の２つのループに分割する。なお、β＝１の場合は、後半ループｍ２は１回しか実行されないのでループにしなくてもよい。また、ループの分割にあわせて、ループ表１１０の更新を行う。

【００５７】次に、後半ループｍ２が多重ループであるか否か、すなわち、その内側に他のループを含むループであるかを否かを調べる（Ｓ４０７）。その結果、後半ループｍ２が多重ループである場合は（Ｓ４０７：ｙｅｓ）、更なるループ分割を行うため、後半ループｍ２の内側ループを、分割対象ループｍとして、ステップＳ４０３～Ｓ４０７の処理を再度行う。なお、後半ループｍ２が複数の（同階層の）内側ループを有する場合は、一番最後のループを分割対象ループｍとする。

【００５８】一方、後半ループｍ２が多重ループでなければ（Ｓ４０７：ｎｏ）、これ以上、ループ分割をすることはできないので、その時点での前半ループｍ１と後半ループｍ２の間を、プリフェッチ命令を挿入する位置とし（Ｓ４０９）、処理を終了する。

【００５９】次に、上述したコンパイラ１０８の処理を、具体例に基づいて説明する。まず、図５（ａ）に示したソースプログラム（図１２（ａ）と同じソースプログラム）をコンパイルする場合について説明する。

【００６０】図５（ａ）に示したプログラムのコンパイルが指示されると、コンパイラ１０８は、ソースプログラムを読み込み、構文解析処理Ｓ２０１を行った後、ループ解析処理Ｓ２０２を行う。

【００６１】図５（ｂ）は、このループ解析処理Ｓ２０２によって作成されるループ表１１０の構成例を示す図である。同図に示すように、ループ表１１０は、各ループに対して、ループ番号７０１、ループ繰り返し変数７０２、繰り返し回数７０３、内側ループ７０４、直前ループ７０５、直後ループ７０６、予測実行サイクル７０７等の情報を格納する。

【００６２】図５（ａ）のプログラムには、ループが２つあり、外側ループ（６０３～６０９）には、「１」、内側ループ（６０５～６０７）には、「２」というループ番号が付けられている。なお、以下では、ループ番号が「１」のループをループ１、ループ番号が「２」のループをループ２、等と呼ぶ。

【００６３】ループ１のループ繰り返し変数は「ｉ」、繰り返し回数は「Ｎ」、内側ループは、ループ「２」、直前ループ、直後ループは、「なし」である。また、ループ２のループ繰り返し変数は「ｊ」、繰り返し回数は「Ｍ」、内側ループ、直前ループ、直後ループはいずれも「なし」である。

【００６４】予測実行サイクル数７０７には、各ループのループ１回あたりの予測実行サイクル数が格納される。この予測実行サイクル数は、ループ内の命令数と各命令の実行サイクル数、内側ループの繰り返し回数等から計算される。図５の例では、ループ２のループ１回あたりの予測実行サイクルを「ｃ」と見積り、ループ１のループ１回あたりの予測実行サイクルは、ループ２以外の部分の実行サイクル数を無視して、「ｃ＊Ｍ」と計算している。

【００６５】以上のようなループ表の作成を終了すると、続いて、コンパイラ１０８は、まず、ループ１を処理対象ループｎとして取り出し（Ｓ２０４）、ループ１に対してソフトウェアプリフェッチ処理を行う（Ｓ２０５）。

【００６６】そのため、まず、処理対象ループｎであるループ１内の未処理の配列参照Ａ（ｉ）を処理対象配列参照ｓとして取り出す（Ｓ３０２）。また、変数Ｌに、ターゲットマシンのメモリレイテンシを代入する（Ｓ３０３）。

【００６７】次に、ループ１が最内側ループか否かを調

べる（Ｓ３０４）。ループ１は、内側ループとして、ループ２を有し、最内側ループではないので（Ｓ３０４：ｎｏ）、次に、プリフェッチ命令を挿入すべき位置を求める（Ｓ３０７）。

【００６８】そのため、まず、処理対象配列参照ｓであるＡ（ｉ）の直前のループであるループ２を分割対象ループｍとする（Ｓ４０１）。また、ｋにＬを代入する（Ｓ４０２）。そして、分割対象ループｍであるループ２の全体予測実行サイクル数を求める（Ｓ４０３）。この場合、ループ２の繰り返し回数Ｍは、コンパイル時には不明なので、ｄ←∞とする。

【００６９】そして、ｄがｋより大きいか否かを調べる（Ｓ４０４）。この場合、ｄ（＝∞）はｋ（＝Ｌ）より大きいので（Ｓ４０４：ｙｅｓ）、現時点の分割対象ループｍであるループ２の分割を行うため、ループ２のループ１回分の予測実行サイクル数ｃを求め、β＝［Ｌ／ｃ］を計算する（Ｓ４０５）。そして、ループ２を、１～（Ｍ－β）回を実行する前半ループｍ１と（Ｍ－β＋１）～Ｍ回を実行する後半ループｍ２とに分割する（Ｓ４０６）。

【００７０】そして、後半ループｍ２が多重ループであるか否かを調べる（Ｓ４０７）。この場合、後半ループｍ２は多重ループではないので（Ｓ４０７：ｎｏ）、現時点での前半ループｍ１と後半ループｍ２の間をプリフェッチ命令を挿入すべき位置とする（Ｓ４０９）。そして、その位置にプリフェッチ命令「prefetch（Ａ（ｉ））」を挿入する（Ｓ３０８）。

【００７１】図５（ｃ）は、プリフェッチ命令を挿入した後のプログラムをソースコードイメージで示したものである。同図（ｃ）に示すように、同図（ａ）のループ（６０５～６０７）が、始めの（Ｍ－β）回を実行する前半ループ（６２４～６２６）と終わりのβ回を実行する後半ループ（６２８～６３０）の２つに分割されて、その間に、配列参照Ａ（ｉ）のプリフェッチ命令（６２７）が挿入されている。

【００７２】このようにすると、プリフェッチ命令（６２７）とデータ参照（６３１）の間に実行されるループは、β回しか回らず、βは一般に元のループ回数Ｍに比べて小さいので、その間に参照されるデータ量は一般に小さく、プリフェッチしたデータがキャッシュから追い出される可能性は低くなる。また、ループのβ回の実行には、Ｌサイクル、すなわち、データをメモリからプリフェッチをするのに要するのと等しいサイクル数が必要となるので、データを参照するときにプリフェッチデータの到着が間に合っていないということがない。

【００７３】次に、コンパイラ１０８のコンパイル処理を、別の具体例に基づいて説明する。図６（ａ）は、コンパイル処理の対象とされる別のソースプログラムを示す図である。ここでは、配列参照Ａ（ｉ１）（８１１）に対してプリフェッチを行う場合を考える。

【００７４】図６（ｂ）は、同図（ａ）のプログラムに対して、コンパイラ１０８が作成するループ表１１０の構成例を示す図である。同図（ａ）のプログラムにはループが４つあり、ループ（８０２～８１２）がループ１、ループ（８０３～８０７）がループ２、ループ（８０４～８０６）がループ３、ループ（８０８～８１０）がループ４とされる。

【００７５】また、ループ１のループ繰り返し変数は「ｉ１」、繰り返し回数は「１０」、内側ループは、ループ「２」、「４」、直前ループ、直後ループは、「なし」である。ループ２のループ繰り返し変数は「ｉ２」、繰り返し回数は「２０」、内側ループは、ループ「３」、直前ループは「なし」、直後ループは、ループ「４」である。ループ３のループ繰り返し変数は「ｉ３」、繰り返し回数は「３０」、内側ループ、直前ループ、直後ループは、いずれも「なし」である。ループ４のループ繰り返し変数は「ｉ４」、繰り返し回数は「４０」、内側ループ、直後ループはいずれも「なし」、直前ループは、ループ「２」である。

【００７６】また、ループ３及びループ４のループ１回あたりの予測実行サイクルを、それぞれ、「５」及び「１」と見積り、ループ２のループ１回あたりの予測実行サイクルを「１５０」（＝５＊３０）、ループ１のループ１回あたりの予測実行サイクルを、「３０４０」（＝１５０＊２０＋１＊４０）と算出している。

【００７７】このようなループ表１１０の作成を終了すると、コンパイラ１０８は、まず、処理対象ループｎとして、ループ１を取り出し（Ｓ２０４）、ループ１に対してソフトウェアプリフェッチ処理を行う（Ｓ２０５）。

【００７８】そのため、まず、処理対象配列参照ｓとして、Ａ（ｉ１）を取り出す（Ｓ３０２）。また、Ｌにターゲットマシンにおけるメモリレイテンシを代入する（Ｓ３０３）。ここでは、Ｌ＝１００とする。次に、処理対象ループｎであるループ１が最内側ループか否かを調べる（Ｓ３０４）。ループ１は、内側ループとしてループ２，４を含み、最内側ループではないので、続いて、プリフェッチ命令を挿入すべき位置を求める（Ｓ３０７）。

【００７９】そのために、まず、処理対象配列参照ｓであるＡ（ｉ１）の直前のループ、すなわち、ループ４を分割対象ループｍとする（Ｓ４０１）。またｋにＬ（＝１００）を代入する（Ｓ４０２）。

【００８０】次に、分割対象ループｍであるループ４の全体予測実行サイクル数ｄを求める（Ｓ４０３）。ループ４のループ１回分の予測実行サイクルは、「１」で、ループ繰り返し回数が「４０」なので、ｄ←４０（＝１＊４０）となる。

【００８１】次に、ｄがｋより大きいか否かを調べる（Ｓ４０４）。この場合、ｄ（＝４０）は、ｋ（＝１０

0）より小さいので（Ｓ４０４：ｎｏ）、分割対象ループｍを、現在の分割対象ループであるループ４の直前のループ、すなわち、ループ２とし、また、ｋ←６０（＝ｋ－ｄ＝１００－４０）とする（Ｓ４１０）。

【００８２】そして、改めて、現在の分割対象ループｍであるループ２の全体予測実行サイクル数ｄを求める（Ｓ４０３）。ループ２のループ１回分の予測実行サイクルは「１５０」で、ループ繰り返し回数が「２０」なので、ｄ←３０００（＝１５０＊２０）となる。

【００８３】次に、ｄがｋより大きいか否か調べる（Ｓ４０４）。この場合、ｄ（＝３０００）は、ｋ（＝６０）より大きいので、分割対象ループｍであるループ２の分割を行うため、次に、ループ２のループ１回あたりの予測実行サイクル数ｅ（＝１５０）を求め、β＝［ｋ／ｅ］＝［６０／１５０］＝１を計算する（Ｓ４０５）。次に、その結果に基づいて、ループ２を２つのループに分割する（Ｓ４０６）。ループ２の繰り返し回数は２０、β＝１なので、ループ２を１～１９回を実行する前半ループｍ１と、最後の１回を実行する後半ループｍ２に分割する。

【００８４】図７（ａ）は、図６（ａ）に示したプログラムのループ２を、前半ループｍ１と後半ループｍ２に分割した時点でのプログラムをソースコードイメージで示したものである。同図に示すように、元のループ（８０３～８０７）が、前半ループ（９０３～９０７）と、後半ループ（９０８～９１２）に分割されている。

【００８５】図７（ｂ）は、ループ２の分割に伴って、コンパイラ１０８が更新した後のループ表１１０を示す図である。同図（ａ）に示すように、ループ２の分割によって、２つのループが増えているが、ループ（９０８～９１２）がループ５、ループ（９０９～９１１）がループ６とされる。

【００８６】また、ループ５のループ繰り返し変数は「ｉ２」、繰り返し回数は「１」、内側ループは、ループ「６」、直前ループは、ループ「２」、直後ループは、ループ「４」、予測実行サイクルは、「１５０」となる。ループ６のループ繰り返し変数は「ｉ３」、繰り返し回数は「３０」、内側ループ、直前ループ、直後ループは、いずれも「なし」、予測実行サイクルは、「５」となる。

【００８７】更に、ループ１の内側ループに、ループ「５」が追加されており、ループ２の繰り返し回数及び直後ループがそれぞれ、「１９」及びループ「５」に更新され、ループ４の直前ループが、ループ「５」に更新されている。

【００８８】コンパイラ１０８は、ループ２の分割を終了すると、次に、後半ループｍ２であるループ５が多重ループであるか否かを調べる（Ｓ４０７）。この場合、ループ５は、内側ループとして、ループ６を含み、多重ループあるので（Ｓ４０７：ｙｅｓ）、分割対象ループｍを、ループ６に変える（Ｓ４０８）。

【００８９】その後、改めて、分割対象ループｍであるループ６の全体予測実行サイクルを求める（Ｓ４０３）。この場合、１５０（＝３０＊５）であるので、それをｄとする。次に、ｄがｋより大きいか否かを調べる（Ｓ４０４）。この場合、ｄ（＝１５０）は、ｋ（＝６０）より大きいので（Ｓ４０４：ｙｅｓ）、ループ６の分割を行うため、ループ６のループ１回あたりの予測実行サイクル数ｅ（＝５）を求め、β＝［ｋ／ｅ］＝［６０／５］＝１２を計算する（Ｓ４０５）。

【００９０】そして、ループ６を最後のβ回のみを実行する後半ループｍ２と、その前までの繰り返しを実行する前半ループｍ１とに分割する（Ｓ４０６）。ループ６の繰り返し回数は３０なので、ループ６を、１～１８（＝３０－１２）回実行する前半ループｍ１と、１９～３０回実行する後半ループｍ２に分割する。続いて、後半ループｍ２が多重ループか否かを調べると（Ｓ４０７）、そうではないので（Ｓ４０７：ｎｏ）、現時点での前半ループｍ１と後半ループｍ２の間をプリフェッチ命令を挿入すべき位置とし（Ｓ４０９）、その位置に、処理対象配列参照ｓであるＡ（ｉ１）についてのプリフェッチ命令を挿入する（Ｓ３０８）。

【００９１】図８は、プリフェッチ命令を挿入した時点でのプログラムをソースコードイメージで示したものである。同図に示すように、前半ループ（１００９～１０１１）と後半ループ（１０１３～１０１５）の間にプリフェッチ命令（１０１２）が挿入されている。なお、この場合、ループ（１００８～１０１６）は１回しか回らないので、前述したように、ここはループにしなくてもよい。

【００９２】図８のプログラムでは、配列要素Ａ（ｉ１）のプリフェッチ命令（１０１２）と配列要素Ａ（ｉ１）の参照（１０２０）の間で、ループ（１０１３～１０１５）とループ（１０１７～１０１９）が実行される。ループ（１０１３～１０１５）の予測実行サイクル数は、５＊（３０－１９＋１）＝６０であり、ループ（１０１７～１０１９）の予測実行サイクル数は、１＊４０＝４０である。つまり、合計１００サイクルであり、メモリレイテンシＬと等しくなる。従って、データの参照にちょうど間に合うように、プリフェッチ命令が発行されることになる。

【００９３】以上の実施形態では、プリフェッチ命令を挿入するためにループを分割していたが、ループを分割せず、ループ中に条件付きのプリフェッチ命令を挿入するようにしてもよい。

【００９４】図９は、図５（ａ）に示したプログラムにおいて、ループを分割してプリフェッチ命令を挿入するかわりに、ループ中に条件付きのプリフェッチ命令実行コード（１１０６～１１０７）を挿入した場合のプログラムをソースコードイメージで示した図である。この場

合、制御変数ｊが（Ｍ－β＋１）と等しいとき、すなわち最後からβ番めの繰り返しのときにＡ（ｉ）のプリフェッチ命令が実行され、プリフェッチ命令が実行されるタイミングは、図５（ｃ）の場合と同じになる。このような処理を行うには、例えば、図４のステップＳ４０６でループを分割するかわりに、プリフェッチ命令を発行すべきループｍの繰り返し回数Ｎ－β＋１を記憶しておいて、その後、その条件に該当するときに処理対象配列参照ｓのプリフェッチ命令を実行するようなコードを挿入するようにすればよい。

【００９５】また、以上の実施形態では、メモリ参照がループ内にある場合を対象にしていたが、ループ内にない場合に適用することもできる。図１０は、プログラム手続き「subroutine f(A,B,i)」に対して、本発明を適用した例を示す図である。

【００９６】ここでは、図１０（ａ）に示したプログラム手続き内の配列参照Ａ（ｉ）（１２０７）に対して、同図（ｂ）のように直前のループを分割してプリフェッチ命令（１２１５）を挿入している。このような処理によって、メモリレイテンシ分だけ前にプリフェッチ命令を実行することができる。このような処理を行うには、図２のステップＳ２０３～Ｓ２０４でループを取り出すとき、プログラム手続き全体を１つの仮想的なループと考えて処理すればよい。

【００９７】

【発明の効果】以上詳細に説明したように、本発明によれば、最内側ループ中にないデータの参照に対して、プリフェッチしたデータが使用される前にキャッシュから追い出されることに起因するキャッシュミスを減少させ、プログラム実行の高速化を図ることができる。

【図面の簡単な説明】

【図１】　本発明によるコンパイラが稼動する計算機システムの構成を示す図である。

【図２】　コンパイラ１０８のコンパイル処理の流れを示すフローチャートである。

【図３】　ソフトウェアプリフェッチ処理の流れを示す図である。

【図４】　プリフェッチ命令を挿入すべき位置を求める処理の流れを示す図である。

【図５】　本発明によるソフトウェアプリフェッチ処理の例を示す図である

【図６】　コンパイル処理の対象とされるプログラムの例とそのループ表を示す図である。

【図７】　ループ分割をした時点でのプログラムのソースコードイメージとそのループ表を示す図である。

【図８】　プリフェッチ命令を挿入した時点でのプログラムをソースコードイメージで示した図である。

【図９】　ループ中に条件付きのプリフェッチ命令実行コードを挿入した場合のプログラムをソースコードイメージで示した図である。

【図１０】　プログラム手続き内の配列参照に対するソフトウェアプリフェッチ処理の例を示す図である。

【図１１】　従来のソフトウェアプリフェッチ方法の例を示す図である。

【図１２】　従来のソフトウェアプリフェッチ方法の例を示す図である

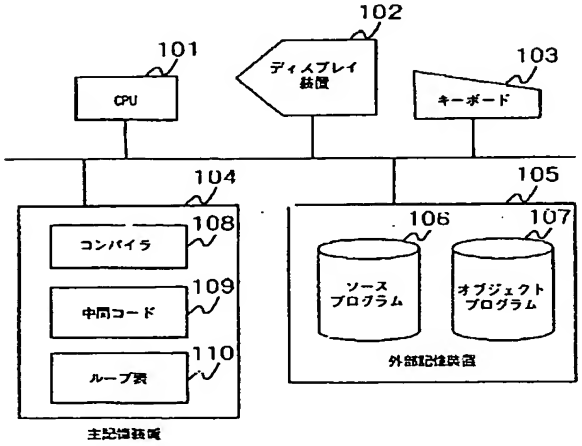
【符号の説明】

１０８　コンパイラ

Ｓ２０５　ソフトウェアプリフェッチ処理

【図１】

図1



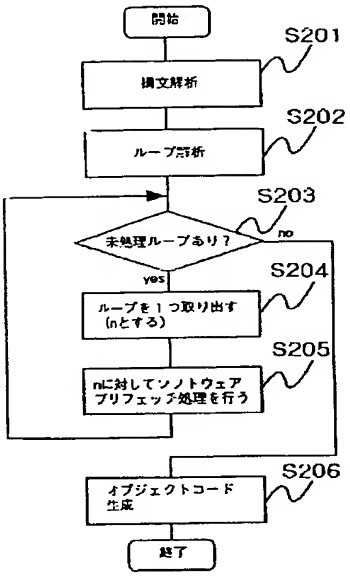
【図８】

図8

```
real A(N)                          (1001)
do i1=1,10                         (1002)
   do i2=1,19                      (1003)
     do i3=1,30                    (1004)
       ...                         (1005)
     enddo                         (1006)
   enddo                           (1007)
   do i2=20,20                     (1008)
     do i3=1,18                    (1009)
       ...                         (1010)
     enddo                         (1011)
     prefetch(A(i1))               (1012)
     do i3=19,30                   (1013)
       ...                         (1014)
     enddo                         (1015)
   enddo                           (1016)
   do i4=1,40                      (1017)
     ...                           (1018)
   enddo                           (1019)
   ...=A(i1)                       (1020)
enddo                              (1021)
```

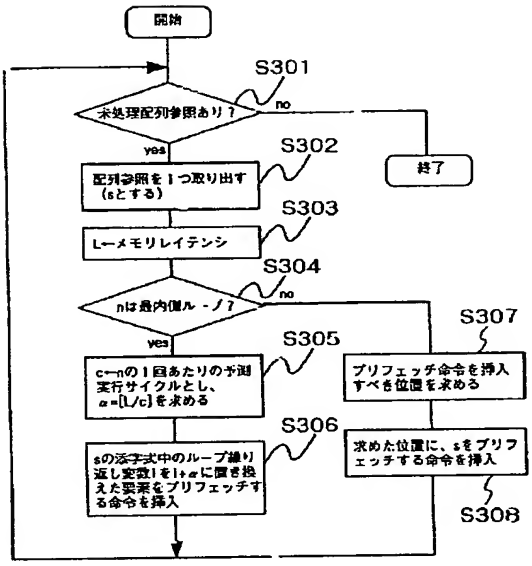
【図2】

図2



【図3】

図3



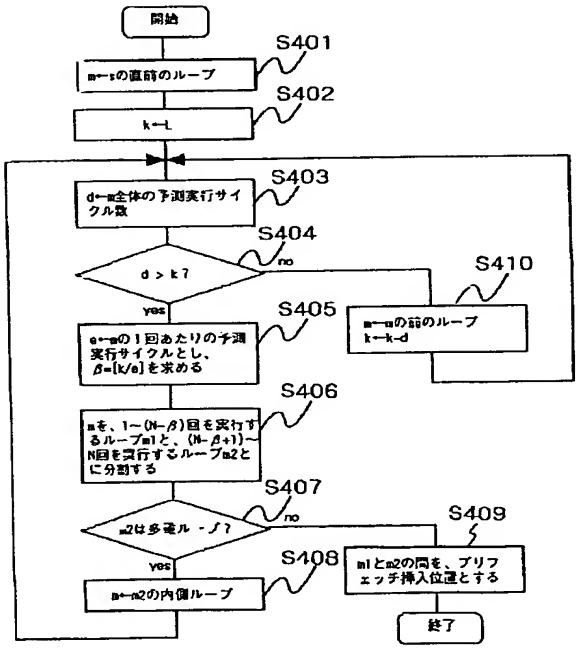
【図9】

図9

```
real A(N),B(M,N)          (1101)
s=0                       (1102)
do i=1,N                  (1103)
  t=0                     (1104)
  do j=1,M                (1105)
    if j=M-β+1 then       (1106)
      prefetch(A(i))      (1107)
    t=t+B(j,i)            (1108)
  enddo                   (1109)
  s=t+A(i)                (1110)
enddo                     (1111)
```

【図4】

図4



【図5】

図5

```
real A(N),B(M,N)          (601)
s=0                       (602)
do i=1,N                  (603)
  t=0                     (604)
  do j=1,M                (605)
    t=t+B(j,i)            (606)
  enddo                   (607)
  s=t+A(i)                (608)
enddo                     (609)
```

(a)

701 ループ番号	702 ループ繰り返し変数	703 繰り返し回数	704 内側ループ	705 直前ループ	706 直後ループ	707 予測実行サイクル
1	i	N	2	なし	なし	c*M
2	j	M	なし	なし	なし	c

(b)

```
real A(N),B(M,N)          (620)
s=0                       (621)
do i=1,N                  (622)
  t=0                     (623)
  do j=1,M-β              (624)
    t=t+B(j,i)            (625)
  enddo                   (626)
  prefetch(A(i))          (627)
  do j=M-β+1,M            (628)
    t=t+B(j,i)            (629)
  enddo                   (630)
  s=t+A(i)                (631)
enddo                     (632)
```

(c)

【図6】

図6

```
real A(N)                (801)
do i1=1,10               (802)
   do i2=1,20            (803)
     do i3=1,30          (804)
       ...               (805)
     enddo               (806)
   enddo                 (807)
   do i4=1,40            (808)
     ...                 (809)
   enddo                 (810)
   ...=A(i1)             (811)
enddo                    (812)
```

(a)

701 702 703 704 705 706 707

ループ番号	ループ繰り返し変数	繰り返し回数	内側ループ	直前ループ	直後ループ	予測実行サイクル
1	i1	10	2,4	なし	なし	3040
2	i2	20	3	なし	4	150
3	i3	30	なし	なし	なし	5
4	i4	40	なし	2	なし	1

(b)

【図7】

図7

```
real A(N)                (901)
do i1=1,10               (902)
   do i2=1,19            (903)
     do i3=1,30          (904)
       ...               (905)
     enddo               (906)
   enddo                 (907)
   do i2=20,20           (908)
     do i3=1,30          (909)
       ...               (910)
     enddo               (911)
   enddo                 (912)
   do i4=1,40            (913)
     ...                 (914)
   enddo                 (915)
   ...=A(i1)             (916)
enddo                    (917)
```

(a)

701 702 703 704 705 706 707

ループ番号	ループ繰り返し変数	繰り返し回数	内側ループ	直前ループ	直後ループ	予測実行サイクル
1	i1	10	2,4,5	なし	なし	3040
2	i2	19	3	なし	5	150
3	i3	30	なし	なし	なし	5
4	i4	40	なし	5	なし	1
5	i2	1	6	2	4	150
6	i3	30	なし	なし	なし	5

(b)

【図10】

図10

```
subroutine f(A,B,i)      (1201)
real A(N),B(M,N)         (1202)
t=0                      (1203)
do j=1,M                 (1204)
  t=t+B(j,i)             (1205)
enddo                    (1206)
t=t+A(i)                 (1207)
end                      (1208)
```

(a)

```
subroutine f(A,B,i)      (1209)
real A(N),B(M,N)         (1210)
t=0                      (1211)
do j=1,M-β               (1212)
  t=t+B(j,i)             (1213)
enddo                    (1214)
prefetch(A(i))           (1215)
do j=M-β+1,M             (1216)
  t=t+B(j,i)             (1217)
enddo                    (1218)
t=t+A(i)                 (1219)
end                      (1220)
```

(b)

【図11】

図11

```
real A(N)                (501)
s=0                      (502)
do i=1,N                 (503)
  s=s+A(i)               (504)
enddo                    (505)
```

(a)

```
real A(N)                (506)
s=0                      (507)
do i=1,N                 (508)
  prefetch(A(i+α))       (509)
  s=s+A(i)               (510)
enddo                    (511)
```

(b)

【図12】

図12

```
real A(N),B(M,N)         (601)
s=0                      (602)
do i=1,N                 (603)
  t=0                    (604)
  do j=1,M               (605)
    t=t+B(j,i)           (606)
  enddo                  (607)
  s=t+A(i)               (608)
enddo                    (609)
```

(a)

```
real A(N),B(M,N)         (610)
s=0                      (611)
do i=1,N                 (612)
  t=0                    (613)
  do j=1,M               (614)
    t=t+B(j,i)           (615)
  enddo                  (616)
  prefetch(A(i+α))       (617)
  s=t+A(i)               (618)
enddo                    (619)
```

(b)